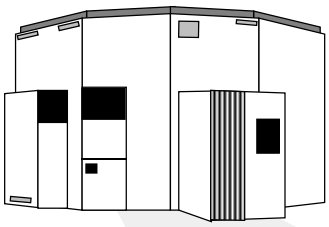
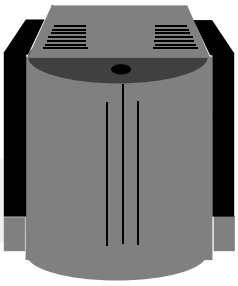
Naval Research Laboratory

Washington, DC 20375-5320





CGI Programming in Perl

Course Notes

Instructor: Michael G. Vonk
Center for Computational Science
(202)767-3884
michael.vonk@nrl.navy.mil

CGI Programming in Perl

1. Introduction

Perl is an easy to use, public domain scripting language that is available for all the major operating systems. For these reasons it has become the "language of choice" for Web programming. Topics covered in this class include:

- the Common Gateway Interface (CGI)
- configuring the Web server to allow CGI
- CGI input and output
- generating dynamic Web pages
- processing form results
- security
- server side includes

This class expands on material covered in the "Web Publishing" and "Programming in Perl" classes. Numerous public domain and commercial Web servers are available, including:

- NCSA httpd (UNIX)
- Personal Web Sharing (included with Macintosh OS 8.0)
- Microsoft Internet Information Server (Windows 95/NT)

All examples from this class, along with pointers to additional information, can be found on the Web at:

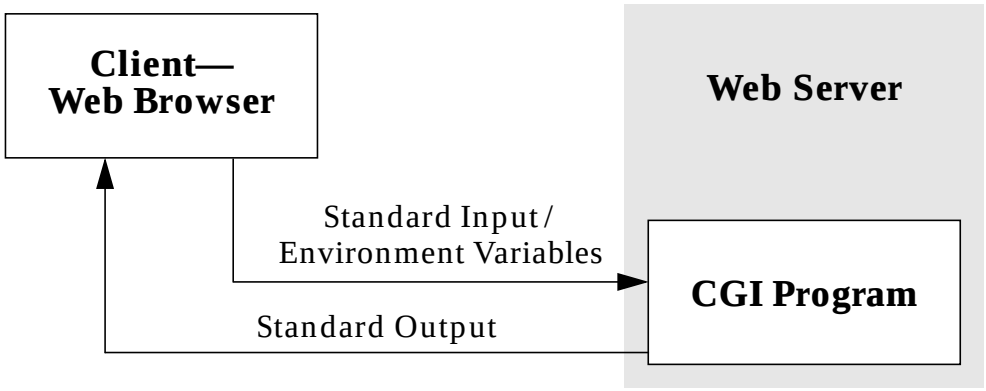
```
http://amp.nrl.navy.mil/code5595/
    ccs-training/cgi
```

2. CGI Overview

CGI scripts allow you to tailor your Web server so that it can communicate with other program on the server, rather than just serving "static" documents. CGI programs allow you to:

- process results from HTML forms
- interact with external programs whose output is not directly accessible on the Web (to interact with a database, for example)
- generate Web pages on the fly

When a browser requests the URL corresponding to CGI script, the server executes the program and sends output back to the browser.



Data is passed to the CGI program via standard input and environment variables. Output from the CGI program is sent back to the browser (typically as HTML code).

Note: CGI programs can be written in any language, although Perl is preferred for its ease of use.

3. Configuring the Web Server

Depending on which Web server you are using, several configuration parameters must be set in order to execute CGI programs.

3.1. NCSA httpd

On a UNIX system running the NCSA httpd Web server, CGI scripts are typically stored in a central location as specified with the ScriptAlias directive in the server resource map file (srm.conf):

```
ScriptAlias  /cgi-bin/  /usr/local/www/cgi-bin/
```

If the user accesses a URL such as:

```
http://yourserver.domain/cgi-bin/hello.cgi
```

then the file "`/usr/local/www/cgi-bin/hello.cgi`" will be executed. Multiple ScriptAlias directives can be specified.

To allow CGI scripts to be placed anywhere on the server, you can use the AddType directive in srm.conf:

```
AddType  application/x-httpd-cgi  .cgi  .pl  .sh
```

In this case, all files with the specified extensions will be executed as CGI programs.

Note: UNIX CGI scripts must be world executable.

The advantage of placing all CGI scripts in one directory is that if poorly written, they can create problems.

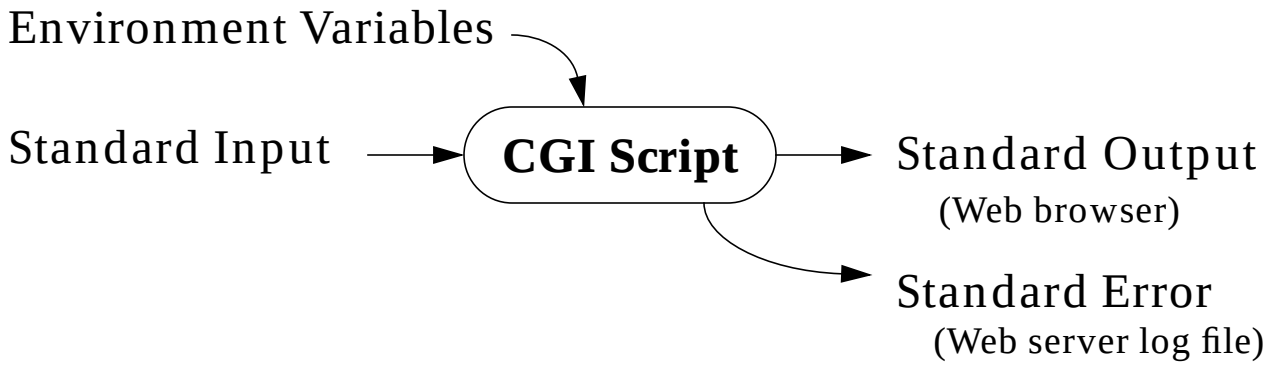
3.2. Personal Web Sharing

Under Macintosh OS 8.0, Web sharing is enabled via the "Web Sharing" control panel. By default, Web pages are stored in the "Web Pages" folder at the root level of the hard disk. CGI scripts are typically created using MacPerl and stored in the "Web Pages" folder or one of its subfolders.

Note: In MacPerl, select the "CGI Script" option in the "Save As..." dialog box.

4. CGI Input and Output

When a CGI script is invoked, the Web server provides it with its own special operating environment, including standard input and output and the ability to generate error messages:



4.1. Input

There are three ways in which the browser can send data to a CGI script:

1. As a "query string" appended to a URL:

   `URL?`*`query-string`*

   This happens as a result of one of three things:

   - a form using the GET method
   - a searchable index (an HTML file that contains an ISINDEX element)*****
   - a server-side image map*****

   The query string is available as an environment variable.

***** Considered obsolete—forms and client-side image maps are much more useful.

2. As standard input from a form using the POST method.
3. As extra path information in the URL. Directory-like information included inside the URL itself immediately following the name of the CGI program.

   URL/*extra-path-info*

   Extra path information is placed in environment variables PATH_INFO and PATH_TRANSLATED.

4.1.1. Environment Variables

Numerous environment variables are accessible from a CGI script and are stored in the associative array %ENV. The following could be used to print a sorted list of all of them:

```
foreach $key (sort keys %ENV) {
    print "$key = $ENV{$key}\n"
}
```

4.1.2. Standard Input

See section on handling form results for examples.

4.2. Output

Standard output from a CGI script is sent back to the browser and can come in many forms:

- HTML code
- plain text
- images (GIF or JPEG)
- audio
- etc.

Normally, the Web browser uses file extensions to determine how to display files it retrieves. For example, HTML files have the extension ".html" and are displayed according to their embedded tags.

Standard output doesn't have a file extension, however. So we output a header to describe the type of data we're returning, followed by a blank line, and then the actual data. For example:

```
print "Content-type: text/html\n";
print "\n"
```

or

```
print "Content-type: image/gif\n";
print "\n"
```

Error messages are generally output to the Web server log files, but can be merged with standard output and sent to the browser using:

```
open(STDERR,">STDOUT");
```

5. Dynamic Web Pages

Web pages can be generated "on the fly" in order to:

- incorporate dynamic information
- format data not directly accessible on the Web
- serve as confirmation notices for forms processing

The generated data is passed back to the client (Web browser) via standard output. The following is the ubiquitous "Hello world" program written as a CGI script:

```
print "Content-type: text/html\n";
print "\n";
print "<html>\n";
print "<head>\n";
print "  <title>CGI Example</title>\n";
print "</head>\n";
print "<body>\n";
print "<h1>CGI Example</h1>\n";
print "Hello, world...<p>\n";
print "</body>\n";
print "</html>\n";
```

Example 1 hello.cgi

This script, placed in the cgi-bin directory (if necessary) and made executable, could be referenced from another page as follows:

```
<a href="/cgi-bin/hello.cgi">Hello</a>
```

6. Web Forms

Creating Web-based forms involves two steps:

- writing an HTML form specification
- writing a script to handle the results of the form

Form results are passed to the form handler script via standard input. An HTML page is typically written back to the browser as a confirmation that the form was submitted.

6.1. HTML Form Specifications

Form specifications are covered in the "Publishing on the Web" class (see its Web companion page). A basic form is as follows:

```
<html>
<h1>Basic Form</h1>

<form method="post"
      action="/cgi-bin/echo-handler.cgi">

Enter your name and favorite color:<p>

Name:  <input type=text name=user><br>
Color: <input type=text name=color><p>

<input type=submit value="Submit">
<input type=reset  value="Reset Form">

</form>
</html>
```

Example 2 basic-form.html

6.2. Handling Form Results

When the user clicks on the "submit" button, form results (a list of field names and their associated values) are passed to the form handler program specified in the "action" attribute. The raw data from the previous form might appear as:

```
user=Michael&color=Blue
```

The process of handling the form results can be summarized as follows:

- Read the raw data
- Split data on "&" into name-value pairs
- For each pair:

  Split pair on "=" into name and value
  Decode name and value
- Add name and its value to associative array
- Process data (save to file, mail, or whatever)
- Write confirmation page back to browser

Certain characters are encoded and must be decoded before they can be used. Spaces are converted into plus signs and other special characters are converted into their hexidecimal values.

7. Security

Everytime a CGI script is invoked, a program is executed on the Web server, potentially leading to security problems. Consider the following innocent looking example:

- an HTML form requests the user to enter a name of a file on the server to be displayed
- the user enters "hello.cgi; rm -rf /"
- the form handling program issues the UNIX cat command to display the file (ie. "cat hello.cgi; rm -rf /")
- the webmaster wonders where all their files have gone

For additional information, see the references on the companion page.

8. Server Side Includes

Server Side Includes (SSIs) are not really CGI programs, but are useful in many ways. They are special directives embedded in HTML documents that can be used to:

- Include files (boilerplate text)
- Display environment variables and file statistics
- Execute external programs and CGI scripts

While SSIs provide many time saving benefits, as compared to writing CGI scripts to create dynamic Web pages, there are two disadvantages:

- performance degradation
- security considerations

Note: SSIs are not supported by all servers—they are supported by the NCSA and Netscape servers, but are not currently supported by the CERN server.

8.1. Configuring the Web Server

Before SSIs can be used, they must be enabled on the server. Using the NCSA httpd Web server, this involves two types of configuration:

1. Modifying the server configuration file (srm.conf) to specify the files that should be parsed for directives:

   ```
   AddType text/x-server-parsed-html .shtml
   ```

   (You could use .html, but then all HTML files would be parsed, thus leading to severely degraded performance.)

2. Setting access control options (access.conf) to indicate what type of SSI directives can be used:

   - To display environment variables and file statistics, use the Includes option
   - To execute external programs, use the Exec option

   For example:

   ```
   Options Include ExecCGI
   ```

   or

   ```
   Options IncludesNoCGI
   ```

8.2. SSI Directive Format

SSI directives are specified as follows:

```
<!--#command parameter="argument"-->
```

Do not put spaces between the `<!--` and `#command`.

8.3. SSI Directives

The following SSI directives are available:

Displaying Environment Variables	All the environment variables available to CGI scripts as well as several others can be displayed: `<!--#echo var="variable"-->`

Including Files	Files in the current directory or in a path relative to the server root can be included:
	`<!--#include file="name"-->`
	`<!--#include virtual="path"-->`

Displaying File Statistics	The size and last modification date of a file can be displayed:
	`<!--#fsize file="name"-->`
	`<!--#flastmod file="name"-->`

Executing External Programs and CGI Scripts	External programs can be executed using: `<!--#exec cmd="command"-->` If environment variables are used in command, they must be prefixed with '$'.

Tailoring SSI Output	The format of error messages, file sizes, and dates can be tailored as follows:
	`<!--#config errmsg="msg"-->`
	`<!--#config sizefmt="fmt"-->`
	`<!--#config timefmt="fmt"-->`

8.4. SSI Security Considerations

Allowing the execution of commands can lead to security problems. For example, some pages allow users to enter HTML code in comment fields for example, and then display this code. The user could then enter:

```
<!--#exec cmd="rm -rf /"-->
```

Well written CGI scripts strip SSI commands from input, thus eliminating the problem.

9. References

Many online and hardcopy references were used in creating this class. Among the best of these are the following:

- "Learning Perl, 2nd Edition"

  by Randal L. Schwartz & Tom Christiansen
  O'Reilly and Associates, Inc.
  ISBN: 1-56592-284-0

- "CGI Programming on the World Wide Web"

  by Shishir Gundavaram
  O'Reilly and Associates, Inc.
  ISBN: 1-56592-168-2

- "Webmaster in a Nutshell"

  by Stephen Spainhour and Valerie Quercia
  O'Reilly and Associates, Inc.
  ISBN: 1-56592-229-8

- the "Perl reference materials" Web page at:

  `http://www.eecs.nwu.edu/perl/perl.html`

- the newsgroup `comp.lang.perl`

10. Summary

CGI scripts greatly increases the effectiveness of your Web site. Using CGI you can generate dynamic, rather than only static, Web pages. Documents not previously viewable within a Web browser, a database for example, can now be formatted and displayed. You can also efficiently process form results.

And all this comes at a very small price—once you understand how CGI scripts handle input and generate output, all the rest is just programming.